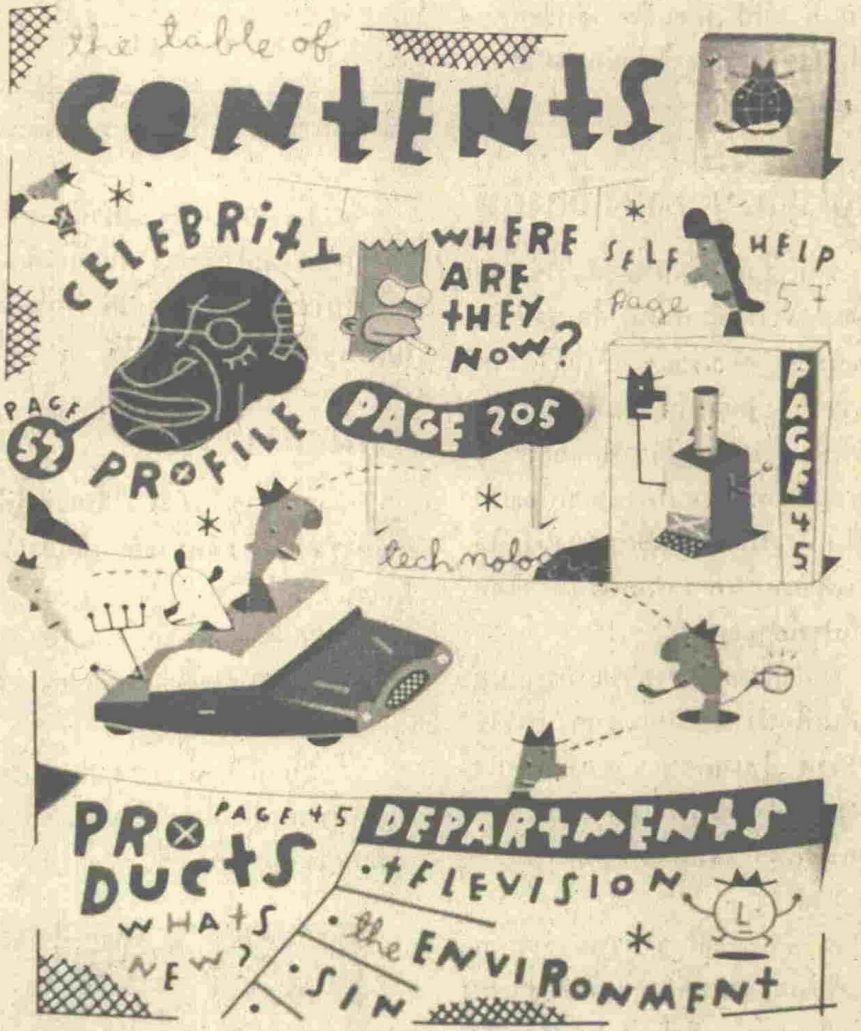
# เทคนิคการสร้างอักขระกราฟิก

***จะทำอย่างไรเมื่อ FONT ไทยที่มีไม่เหมาะกับงานของเรา งั้นมาลองหาทางสร้าง FONT ใหม่ขึ้นมาใช้เองดีกว่า***




จากบทความที่ผู้เขียนได้แสดงการสร้างอักขระกราฟิกในหลายฉบับ เช่น ฉบับที่ 52 และ 62 เป็นต้น ผู้อ่านได้สอบถามถึงวิธีการที่จะสร้างอักขระกราฟิกว่ามีรูปแบบอื่น ๆ อีกหรือไม่ เพื่อสนองตอบคำถามนี้ให้เกิดประโยชน์สูงสุด จึงขอนำมาเขียนถึงในวารสารแทนทั้งนี้เพื่อให้ผู้อ่านท่านอื่น ๆ ได้มีความรู้ไปพร้อมกันด้วย เนื่องจากไม่อยากให้ได้อักขระกราฟิกซ้ำกับที่เคยเขียนมาแล้ว จึงขอยกตัวอย่างจากแฟ้มตัวอักขระที่ใช้อยู่ในราชวิถีเวิร์ดพีซีสำหรับเครื่องพิมพ์ 24 เข็ม ในไฟล์ *.lq1 ซึ่งจะได้รูปแบบการเก็บรหัสดังบทความเรื่องการพิมพ์อักขระกราฟิกตอนที่ 2 ในฉบับที่ 64 รูปที่ 11 หน้า 286 อันประกอบด้วย 108 ไบต์ต่อหนึ่งอักขระ รวมทั้งหมด 224 อักขระจึงทำให้ไฟล์ยาว 108×224=24192 ไบต์ โดยอาศัยข้อมูลจากไฟล์ font1.lq1 ทำให้ได้อักขระกราฟิกเป็นตัวขนาดใหญ่คือ 36×24 จุด นำมาแสดงผลบนจอภาพด้วยเทคนิคสำคัญ 3 วิธี คือ

1. ใช้การลงจุดจำนวน 108×8=864 จุด โดยอาศัยคำสั่ง

```
putpixel(x,y,color);
```

2. ใช้การกำหนดรูปแบบเส้นตรงและการลากเส้น จำนวน 108 เส้น โดยอาศัยคำสั่ง 2 คำสั่งคือ

```
setlinestyle (4,userpattern, 1)
และ line (X1, Y1, X2, Y2);
```

3. ใช้การกำหนดรูปแบบพื้นและการวาดรูปสี่เหลี่ยมจำนวนเพียง 12 รูป สี่เหลี่ยมโดยอาศัย 2 คำสั่งเช่นกัน คือ

```
setfillpattern(userpatterntype,color);
และ bar(X1,Y1,X2,Y2);
```

ตามแนวทางนี้ผู้อ่านลองนึกดูก่อนว่าวิธีใดจะแสดงอักขระบนจอภาพได้รวดเร็วกว่ากัน โดยวิธีเหล่านี้ท่านสามารถนำไปใช้ได้กับข้อมูลของอักขระ

ขนาดปกติอื่น ๆ เช่น ไฟล์ข้อมูลอักขระของ CU-writer แต่ท่านต้องหาให้ได้ก่อนว่าอักขระแต่ละตัวนั้นใช้จำนวนกี่ไบต์ และแก้ไขคำสั่งให้ตรงกับไฟล์ข้อมูลก็จะได้อักขระกราฟิกขนาดต่าง ๆ มาใช้งาน สำหรับฉบับนี้ผู้เขียนมีไฟล์ทั้งสิ้น 6 ไฟล์ โดยแต่ละวิธีประกอบด้วยไฟล์แสดงผลและไฟล์ข้อมูลอักขระอย่างละหนึ่งไฟล์

```
PROGRAM DrawChar_by_arthan_Sattayarak;

uses crt,dos,graph;
type
     PatternChar=array[1..12] of FillpatternType;

var Monoscreen:boolean;
    regs:Registers;  ch,Funckey:char;
    Shape:array[1..24192] of byte;
    ShapeBar:array[0..223] of patternchar;
    k:word;  ColorASCII:byte;
```

รูปที่ 1 ตัวแปรที่กำหนดไว้สำหรับการสร้างอักขระทั้งสามแบบ

## ตัวแปรและชุดคำสั่งเบื้องต้น

ชุดคำสั่งเบื้องต้นที่จะต้องใช้ (แต่ไม่ได้แสดงไว้ในบทความ) คือ ชุดคำสั่ง opengraph; readchr; ซึ่งหาได้จากบทความของผู้เขียนในฉบับที่ผ่าน ๆ มา เพื่อใช้เป็นการนำเข้าสู่กราฟิกและตรวจการกดแป้นพิมพ์ ส่วนตัวแปรเบื้องต้นนั้นเป็น ดังรูปที่ 1 ซึ่งเขียนล่วงหน้าไว้ใช้สำหรับเทคนิค 3 แบบดังกล่าวโดยมีตัวแปรที่สำคัญดังนี้

PatternChar สำหรับเก็บรูปแบบของพื้นเพื่อนำไปใช้กับเทคนิค การใช้คำสั่งวาดรูปสี่เหลี่ยม โดยที่ในหนึ่งอักขระจะต้องวาดรูปสี่เหลี่ยม 12 ครั้ง ข้อมูลชนิดนี้จะใช้หน่วยความจำ 12×8 =96 ไบต์

Monoscreen สำหรับตรวจสอบจอภาพ เพื่อใช้ในชุดคำสั่ง opengraph;

Shape ตัวแปรมิติสำหรับรองรับข้อมูลที่อ่านได้จากไฟล์ font1.lq1 หรือ font1.lin สำหรับการใช้เทคนิคการลงจุดและการลากเส้นตามลำดับ จะใช้หน่วยความจำ 24192 ไบต์

ShapeBar ตัวแปรมิติสำหรับรองรับข้อมูลที่อ่านได้จากไฟล์ font1.bar ที่ใช้ในการวาดรูปสี่เหลี่ยม โดยจะใช้หน่วยความจำ 224×96=21504 ไบต์ ซึ่งน้อยกว่าข้อมูลเดิมในราชวิถี

ColorASCII ตัวแปรกำหนดสีของอักขระ มีค่าเป็น 1 ในกรณีที่เป็นจอโมโนโครมแต่ถ้าเป็นจอสีก็กำหนดค่าไปตามชนิดของจอภาพที่ใช้

## เทคนิคการลงจุด

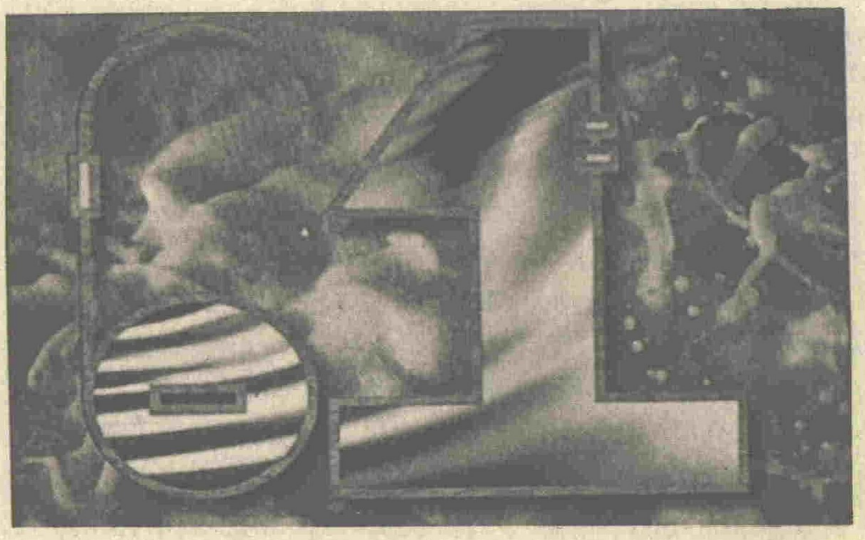
การแสดงอักขระแบบแรกก็คือการลงจุด โดยวิธีที่ตรวจข้อมูลที่อ่านมาได้ดังนี้

– ไบต์ที่ 1,4,7,10,13,...,103,106 เป็นอักขระกลุ่มแรกจำนวน 8 แถวตามแนวราบแถวละ 36 จุด

– ไบต์ที่ 2,5,8,11,14,...,104,107 เป็นอักขระกลุ่มที่สองจำนวน 8 แถวตามแนวราบตรงกลาง

– ไบต์ที่ 3,6,9,12,15,...,105,108 เป็นอักขระกลุ่มที่สองจำนวน 8 แถวตามแนวราบชั้นล่างสุด

โดยจะนำข้อมูลแต่ละไบต์มาตรวจสอบ 8 ครั้งว่ามีบิตใดลงจุดบ้าง ถ้าบิตใดลงจุดก็ให้ใช้คำสั่งลงจุดคือ putpixel ลงตรงจุดที่คำนวณได้ ซึ่งมีผลให้ต้องตรวจกันถึง 108×8=864 ครั้ง การแสดงอาจจะไวขึ้นถ้าตรวจค่าในแต่ละไบต์ว่าเป็นศูนย์หรือไม่ ถ้าใช่ก็ข้ามไปไม่ต้องลงจุด แต่ต้องเลื่อนตำแหน่งของจอภาพให้ถูกต้องด้วย ลำดับของรหัสแอสกีที่บรรจุในไฟล์ font1.lq1 เป็นไปดังรูปที่ 3 หน้า 280 ของฉบับที่ 52 เรื่อง "การทำอักขระกราฟิกด้วย WORD PC" การใช้



```
PROCEDURE LoadFile;
var f:file;
begin
   assign(f,'font1.lq1'); reset(f,1);
    blockread(f,Shape,24192);
   Close(f);
end;

PROCEDURE DrawChar(x,y:word; ASCII:byte);
const BitP:array[0..7] of byte=(128,64,32,16,8,4,2,1);
var yy,k,Yold:word;

begin
  yOld:=y;
 for k:=1 to 108 do
    begin
      for yy:=0 to 7 do
       begin
        if Shape[(ASCII-32)*108+k] and BitP[yy]=BitP[yy] then
                                putpixel(x,y+yy,ColorASCII);
       end;
       inc(y,8);
       if k mod 3=0 then begin inc(x); y:=Yold; end;
    end;
end;
```

รูปที่ 2 ชุดคำสั่งในการอ่านไฟล์ข้อมูลและการแสดงอักขระแบบลงจุด

คำสั่งต่างก็เป็นดังรูปที่ 2 ซึ่งคงไม่ต้องอธิบายเพราะคล้ายกับที่เขียนถึงในฉบับที่ 52 และ 62 จึงขอให้ทำความเข้าใจด้วยตนเอง ชุดคำสั่งในรูปที่ 2 นี้จะเปลี่ยนไปใช้ชุดอื่นแทน ในกรณีที่เราได้เปลี่ยนแปลงเทคนิคการสร้างอักขระไปเป็นแบบอื่น ๆ

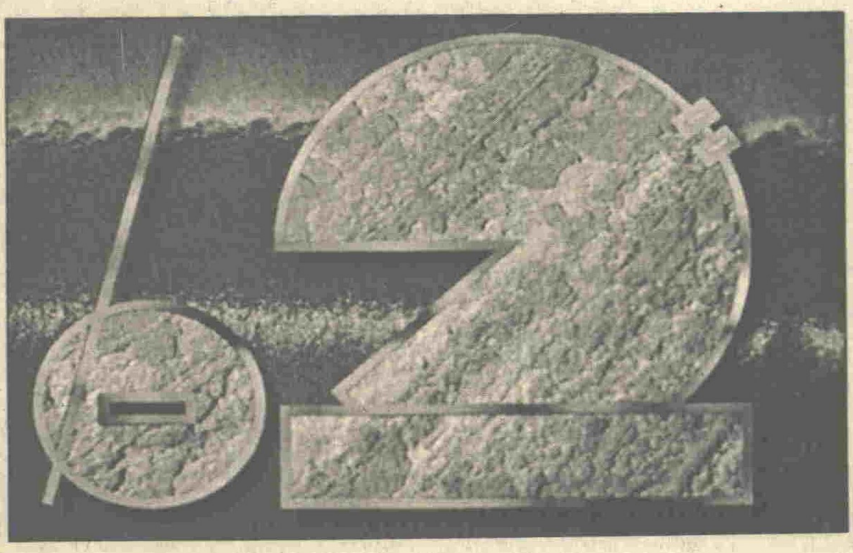
## การแสดงผลตามอักขระวิธี

เพื่อให้การแสดงผลของข้อความที่เรากำหนดตรงกับอักขระวิธีของภาษาไทย จึงเขียนชุดคำสั่งใช้งานคือ OutTextBig ขึ้นมาโดยการกำหนด x, y เป็นตำแหน่งเริ่มต้นแสดงข้อความ และ content เป็นตัวแปรข้อความที่ต้องการแสดง คือ

```
OutTextBig (x,y : word;
Content : string);
```

ซึ่งเป็นดังรูปที่ 3 โดยจะใช้ชุดคำสั่งนี้ในเทคนิคอื่น ๆ ด้วย แต่จะเปลี่ยนค่าตัวแปรให้เหมาะสมเท่านั้น โดยจะมีการกำหนดค่าตัวแปร Xinc และ Yinc ให้เหมาะสมกับการใช้เทคนิคนั้น ๆ อีกที การทำงานของชุดคำสั่งก็จะเริ่มต้นที่การตรวจข้อความเพื่อจัดอักขระให้ถูกต้องตามรูปแบบเสียก่อน เช่น การตรวจสระผสมของ ั ' ให้เป็น ั่ เป็นต้น ซึ่งเป็นช่วงของการวนรอบ (repeat) ส่วนการนำข้อความที่ได้เปลี่ยนแปลงแล้ว (Wordstck) ไปแสดงผลบนจอภาพแบบกราฟิกจะเป็นดังในช่วงถัดมา ซึ่งจะมีการตรวจสอบเฉพาะบางรหัส เช่น

- ตรวจสระ ุ ู (#215, #216) เพื่อเลื่อนตำแหน่งแสดงผลให้ถูกต้อง คือแสดงที่ข้างล่างของอักขระก่อนหน้านี้



```
PROCEDURE OutTextBig(x,y:word; content:string);
const ASCII:array[1..5] of byte=(14,19,23,27,10);
var K,Exit,Xinc,Yinc:byte;  Wordstck:string;
    Ypix:integer;

begin   WordStCk:='';  content:=content+' ';
        k:=1; setcolor(ColorASCII); Xinc:=30; Yinc:=20;
 repeat
   if content[k] in [#149..#159] then
                 WordStCk:=WordStCk+chr(ord(Content[k])-21);
   if content[k] in [#1..#126,#160..#216,#222,#223,#228] then
               WordStCk:=WordStCk+Content[k];
   if content[k] in [#224..#227] then           begin
     if content[k+1]=#207 then WordStCk:=WordStCk+chr(ord(content[k])+6)
       ELSE WordStCk:=WordStCk+Content[k];    end;
   if content[k] in [#217..#221]  then
    begin
     if content[k+1] in [#1..#223] then WordStCk:=WordStCk+Content[k];
     if content[k+1] in [#224..#228] then  begin
      case content[k] of
       #217..#221: Begin
         WordStCk:=WordStCk+chr(ord(content[k+1])+ASCII[ord(content[k])-216]);
       Inc(k);     end;
      end;                                    end;
    end;
       Inc(k);
 until k>Length(Content)-1;
    for k:=1 to length(WordStCk) do
     begin
        ypix:=0;
        if WordStck[k] in [#215,#216] then begin Dec(x,Xinc); ypix:=Yinc; end;
        if WordStck[k] >#216 then begin Dec(x,Xinc); ypix:=-Yinc; end;
        if WordStck[k] in [#210..#212] then
          begin
               DrawChar(x,y+ypix,131);
               DrawChar(x,y-Yinc,ord(WordStck[k])-78);
          end
        Else DrawChar(x,y+ypix,ord(WordStck[k]));
        if WordStck[k]=#207 then Drawchar(x-Yinc,y-Yinc,222);
        if WordStck[k] in [#171..#174] then
               DrawChar(x,y+Yinc,ord(WordStck[k])-36);
        Inc(x,Xinc);
     end;
end;

{====================== DrawChar by PutPixel ===========}

begin
 opengraph;
 loadfile; ColorASCII:=1;
 if pred(Monoscreen) then ColorASCII:=14;
  outtextBig(10,50,'เทคนิคการลงจุด');
  outtextbig(10,100,'จำนวน 108x8 = 864 จุด');
  outtextbig(10,150,'จะได้อักขระ 1 ตัว');
  readchr;
 closegraph;
end.
```

**รูปที่ 3 ชุดคำสั่งในการแสดงผลให้ถูกอักขระวิธีและตัวอย่างการเรียกใช้**

– ตรวจสระบนทั้งหมด (รหัสมากกว่า #216) เพื่อให้แสดงผลเหนืออักขระ

– ตรวจสระ ไ ไ (#210..#212) เพื่อให้แสดงอักขระให้เป็นแบบต่อหัวให้สวยงาม

– ตรวจสระ ำ (#207) เพื่อให้แสดงตัวอักขระ ให้มีตำแหน่งสูงขึ้น

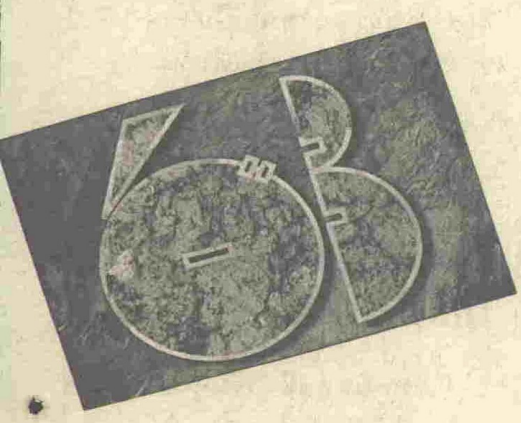
– ตรวจอักขระ ญ ฎ ฏ ฐ (#171

...#174) เพื่อให้แสดงส่วนที่เป็นเชิงของอักขระนั้น ๆ

โดยในตอนท้ายได้ลองแสดงตัวอย่างของข้อความให้ปรากฏบนจอภาพ และจะได้ผลลัพธ์ดังรูปที่ 4 ซึ่งใช้คำสั่ง copy จากจอภาพมาลงกระดาษอีกที โปรแกรมทั้งหมดที่กล่าวมานี้อยู่ในไฟล์ชื่อ ASCII1.PAS

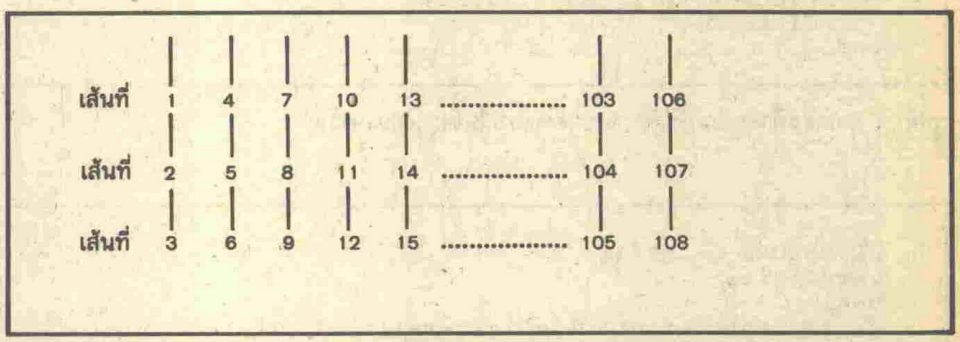
## เทคนิคการลากเส้นตรง

การสร้างอักขระจากข้อมูลที่เก็บอีกวิธีหนึ่งคือ การกำหนดรูปแบบของเส้นให้ตรงกับข้อมูลที่มี และใช้คำสั่งลากเส้น (ต้องลาก 16 จุด) เพราะข้อมูลชนิดนี้ปาสคาลกำหนดมาเป็นตัวแปรชนิด word แม้ข้อมูลในไฟล์จะเป็นข้อมูลชนิดไบต์ก็ตาม ข้อมูลที่เก็บในราชวิถีเวิร์ดพีซีนี้เราสามารถจะลากเส้นให้เป็นตัวอักขระบนจอภาพได้ตามแนวคิดตามรูปที่ 5 ซึ่งจะเห็นว่าเรียงลำดับตามที่ข้อมูลเก็บไว้อยู่แล้ว แต่ผลปรากฏว่าเรายังไม่สามารถนำข้อมูลในไฟล์ font1.lq1 มาใช้ได้เพราะเราจะต้องลากเส้นในแนวดิ่ง ซึ่งถ้าใช้ของเดิมจะได้เส้นที่กลับหัวจึงต้องแก้ไขข้อมูลนี้เสียใหม่ให้มาอยู่ในไฟล์ชื่อ font1.lin การเปลี่ยน

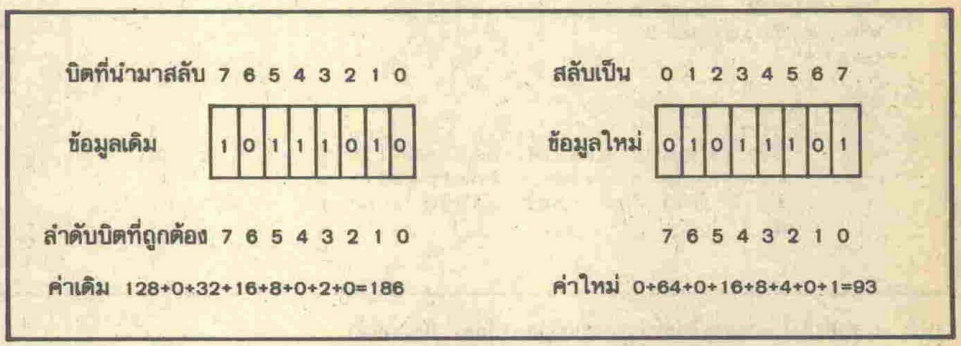


เทคนิคการลงจุด
จำนวน 108x8 = 864 จุด
จะได้อักขระ 1 ตัว

รูปที่ 4 ตัวอย่างที่ปรากฏบนจอภาพของอักขระแบบลงจุด




รูปที่ 5 แสดงตำแหน่งของเส้นที่จะลากเทียบกับข้อมูลในไฟล์อักขระราชวิถีเวิร์ดพีซี




รูปที่ 6 แสดงการสลับข้อมูลเพื่อให้ได้ผลของการลากเส้นตรงกับอักขระที่เก็บไว้

ข้อมูลนี้ก็ทำได้โดยการสลับที่ข้อมูลใหม่ดังตัวอย่างในรูปที่ 6 โดยเราอาจจะทำการเปลี่ยนตอนอ่านไฟล์เข้ามาก็ได้ แต่เพื่อความรวดเร็วในการแสดงผล เราจึงจะเปลี่ยนแปลงข้อมูลใหม่ แล้วเก็บลงไฟล์ไว้ก่อน ซึ่งชุดคำสั่งในเรื่องนี้คือ รูปที่ 7 ชื่อ EditLineMode; โดยชุดคำสั่งนี้อยู่ในไฟล์ชื่อ ASCII2.PAS ด้วยแล้ว โดยไฟล์ font1.lin นี้ยังใช้หน่วยความจำเท่าเดิมคือ 24192 ไบต์ ดังนั้นจึงยังนำมาไว้ที่ตัวแปรเดิมได้ (ตัวแปรมิติชื่อ Shape) ส่วนการนำข้อมูลนั้นมาแสดงผลบนจอภาพจะมีวิธีการอยู่ในชุดคำสั่งของรูปที่ 8 ชุดคำสั่งนี้ให้นำไปแทนในชุดคำสั่งบนรูปที่ 2 ก็จะได้ไฟล์ ASCII2.PAS ซึ่งการแสดงอักขระ

```
PROCEDURE EDitLineMode;
var k:word;  f:file;
    Function SwapBit(B:byte) : byte;
      begin
        SwapBit:=B and 1 shl 7+B and 2 shl 5+B and 4 shl 3+
                 B and 8 shl 1+B and 16 shr 1+B and 32 shr 3+
                 B and 64 shr 5+B and 128 shr 7;
      end;
begin
   assign(f,'font1.lq1'); reset(f,1);
    blockread(f,Shape,24192);
   Close(f);
  for k:=1 to 24192 do
      Shape[k]:=Swapbit(Shape[k]);
   assign(f,'font1.lin');
    rewrite(f); reset(f,1);
    blockwrite(f,Shape,24192);
    Close(f);
end;
```

รูปที่ 7 ชุดคำสั่งเพื่อดัดแปลงไฟล์ข้อมูลแบบลงจุดมาเป็นข้อมูลแบบลากเส้น

```
PROCEDURE LoadFile;
var f:file;
begin
   assign(f,'font1.lin'); reset(f,1);
    blockread(f,Shape,24192);
   Close(f);
end;

PROCEDURE DrawChar(x,y:word; ASCII:byte);
var k,Yold:word;
begin
  yOld:=y;
 for k:=1 to 108 do
    begin
      setlinestyle(4,Shape[(ASCII-32)*108+k],1);
      line(x,y,x,y+15); inc(y,8);
      if k mod 3=0 then begin inc(x); y:=Yold; end;
    end;
end;
```

รูปที่ 8 ชุดคำสั่งในการอ่านไฟล์ข้อมูลและการแสดงอักขระแบบลากเส้น

ด้วยเทคนิคการลากเส้นตามต้องการ โดยพอจะอธิบายรายละเอียดของรูปที่ 8 ได้ว่า คำสั่งอ่านไฟล์ยังคงเป็นแบบเดิม ทั้งนี้เพราะจำนวนข้อมูลเท่ากันเพียงแต่เป็นข้อมูลคนละชุดเท่านั้น โดยมีการแสดงอักขระตามชุดคำสั่ง Drawchar เป็นดังนี้

1. กำหนดค่า Yold:=y เพื่อใช้เป็นจุดเริ่มต้นของเส้นที่ 4, 7....,103, 106

2. กำหนดการลากเส้นจากค่า k มีค่าตั้งแต่ 1 ถึง 108 แล้วทำตามวนรอบนี้

3. กำหนดรูปแบบของเส้นตามคำสั่ง

```
setlinestyle(4,shape [(ASCII – 32)
        * 108 + k],1);
```

โดยความหมายของตัวแปร (4) คือรหัสให้รับรู้ว่าเป็นการใช้รูปแบบเส้นที่กำหนดขึ้นโดยผู้ใช้ Shape [(ASCII-32)*108+k] คือค่าที่อ่านได้ตามข้อมูลของตัวอักขระที่ต้องการเช่น A ซึ่งมีรหัสแอสกีเป็น 65 ก็จะเริ่มข้อมูลที่ shape [(65−32)* 108+1] = Shape [33*108+1]=Shape [3565] ไปจนถึงข้อมูลที่ Shap [[65−32]*108+108] =Shape [33*108+108]=Shape[3672] เป็นต้น (1) คือรหัสให้รู้ว่าการลากเส้นนี้จะเป็นการลากเส้นบาง ๆ เพียงเส้นเดียว ขอให้ลองเปลี่ยนเป็นตัวเลขนี้ให้เป็นเลข (3) แล้วสังเกตผลว่าจะได้ตัวอักขระที่เปลี่ยนแปลงไปอย่างไร

4. ลากเส้นตรง 1 เส้นในแนวดิ่งจำนวน 16 จุดแต่จะเกิดผลเพียง 8 ผล (เพราะข้อมูลกำหนดเป็นไบต์ จึงได้การลงจุดเพียง 8 บิตเท่านั้น)

5. ขยับค่า y ลง 8 จุด

6. ถ้าเป็นเส้นที่ 3, 6, 9,....ก็ให้กำหนดค่า x และ y ใหม่

## เทคนิคการวาดรูปสี่เหลี่ยม

เทคนิคสุดท้ายคือการวาดรูปสี่เหลี่ยมให้เกิดอักขระขึ้นมา ซึ่งฟังดู

ไม่น่าเป็นไปได้ว่าจะเกิดอักขระขึ้นมาได้ การทำแบบนี้เราต้องกำหนดข้อมูลให้เป็นรูปแบบชนิด fillpatterntype ซึ่งจะเก็บข้อมูลขนาด 8 ไบต์ต่อหนึ่งชุด และเพื่อให้รูปสี่เหลี่ยมขนาด 8×8 จุด ลงบนจอภาพได้ ก่อนอื่นต้องดัดแปลงข้อมูลของเดิมที่มีอยู่มาเป็นข้อมูลตามที่เราต้องการก่อนโดยจะใช้ข้อมูล 1 ชุดต่อหนึ่งอักขระดังรูปที่ 9 จะเห็นว่าในแนวนอนเราต้องใช้จุด 4×8=32 จุดแต่ข้อมูลของราชวิถีตามแนวนอนนั้นมีอยู่ 36 จุด (ขอให้ดูรูปที่ 11 หน้า 286 ในฉบับที่ 64 ประกอบ) ดังนั้นเราจึงจำเป็นต้องตัดข้อมูลบางส่วนทิ้งไป ในที่นี้ผู้เขียนเลือกตัดข้อมูลลำดับที่ 1-12 ทิ้ง โดยใช้ข้อมูลที่ 13 เป็นต้นไป ส่วนในแนวดิ่งนั้นข้อมูลมี 24 แถวก็นำมาใช้ทั้งหมดเลย การดัดแปลงนั้นเราจะต้องนำข้อมูล 8 จุดในแนวนอนมาเป็นข้อมูล 1 ไบต์ แล้วนำข้อมูล 8 แถวมากำหนดเป็นข้อมูลชนิด Fillpatterntype เพื่อให้ได้จุดจำนวน 8×8 จุดตามต้องการ ตัวอย่างเช่น ข้อมูลที่ 1 มาจากข้อมูลไบต์ที่ 13, 16, 19, 22, 25, 28, 31, 34 จำนวน 8 ไบต์

| | | | |
|---|---|---|---|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

รูปที่ 9 แสดงลำดับข้อมูลชนิด fillpatterntype ที่จะนำไปวาดรูปสี่เหลี่ยม

บิตที่ 7 ของทุกไบต์เป็นข้อมูลลำดับที่ 1 (ของข้อมูลชนิด Fillpatterntype)

บิตที่ 6 ของทุกไบต์เป็นข้อมูลลำดับที่ 2

บิตที่ 5 ของทุกไบต์เป็นข้อมูลลำดับที่ 3

บิตที่ 4 ของทุกไบต์เป็นข้อมูลลำดับที่ 4

บิตที่ 3 ของทุกไบต์เป็นข้อมูลลำดับที่ 5

บิตที่ 2 ของทุกไบต์เป็นข้อมูลลำดับที่ 6

บิตที่ 1 ของทุกไบต์เป็นข้อมูลลำดับที่ 7

บิตที่ 0 ของทุกไบต์เป็นข้อมูลลำดับที่ 8

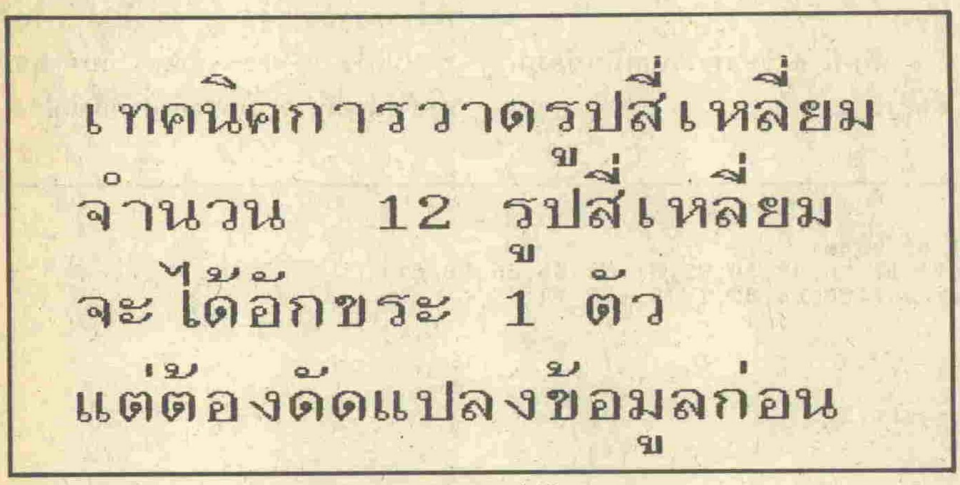
ข้อมูลอื่นก็พิจารณาแบบเดียวกันตามลำดับที่กำหนดไว้ในรูปที่ 9 ซึ่งจะได้ข้อมูลมา 12 ชุดต่ออักขระ ทำให้ใช้หน่วยความจำ 12×8=96 ไบต์ หรือทั้งหมดใช้ 96×224=21504 ไบต์ ชุดคำสั่งในรูปที่ 10 เป็นชุดคำสั่งที่ดัดแปลง

```
PROCEDURE EditBarMode;
const ByteStart:array[1..12] of byte=
                          (13,14,15,37,38,39,61,62,63,85,86,87);
      BitCk:array[0..7] of byte=(128,64,32,16,8,4,2,1);

var f:file;
    ASCII,B:word; BB,Bit,no:byte;
begin
   assign(f,'font1.lq1'); reset(f,1);
    blockread(f,Shape,24192);
   Close(f);
for ASCII:=0 to 223 do
 begin
   for no:=1 to 12 do
    begin
     for bit:=1 to 8 do
      begin
          B:=ASCII*108+ByteStart[no]; ShapeBar[ASCII][no][bit]:=0;
       for bb:=0 to 7 do
        begin
         if Shape[B] and Bitck[bit-1]=Bitck[bit-1] then
            ShapeBar[ASCII][no][bit]:=ShapeBar[ASCII][no][bit]+Bitck[bb];
         inc(B,3);
        end;
      end;
    end;
  end;
   assign(f,'font1.bar');
    rewrite(f); reset(f,1);
    blockwrite(f,ShapeBar,21504);
   Close(f);
end;
```

รูปที่ 10 ชุดคำสั่งเพื่อดัดแปลงไฟล์ข้อมูลแบบลงจุดมาเป็นข้อมูลแบบวาดรูปสี่เหลี่ยม

```
PROCEDURE LoadFile;
var f:file;
begin
   assign(f,'font1.bar'); reset(f,1);
    blockread(f,ShapeBar,21504);
   Close(f);
end;

PROCEDURE DrawChar(x,y:word; ASCII:byte);
var k,yold:word;
begin
  yOld:=y;
 for k:=1 to 12 do
    begin
      setfillpattern(ShapeBar[ASCII-32][k],ColorASCII);
      bar(x,y,x+7,y+7); inc(y,8);
      if k mod 3=0 then begin y:=yOld; inc(x,8); end;
    end;
end;
```

รูปที่ 11 ชุดคำสั่งในการอ่านไฟล์ข้อมูลและการแสดงอักขระแบบวาดรูปสี่เหลี่ยม




รูปที่ 12 ภาพตัวอย่างที่ได้จากการสร้างอักขระแบบวาดรูปสี่เหลี่ยม

ไฟล์ font1.lq1 ไปเป็นไฟล์ font1.bar เพื่อใช้ในการวาดรูปอักขระด้วยการวาดรูปสี่เหลี่ยมต่อไป ส่วนการแสดงผลอักขระบนจอภาพก็ใช้ชุดคำสั่งในรูปที่ 11 ซึ่งคงทำความเข้าใจกับชุดคำสั่งนี้ได้ไม่ยากนัก เพราะคำสั่งสั้นมากการวาดอักขระด้วยคำสั่งวาดรูปสี่เหลี่ยมนั้นมีข้อระวังคือ จุดเริ่มต้นของการวาดรูปสี่เหลี่ยมต้องเป็นจุดที่หารด้วย 8 ลงตัว ทั้งนี้รวมทั้งการเปลี่ยนแปลงค่า x หรือ y ด้วย ต้องเพิ่มหรือลดด้วยค่าที่หารด้วย 8 ลงตัวไปด้วยเช่นกัน (ถ้าไม่เชื่อให้ลองเปลี่ยนค่าดูก็ได้ก็จะเห็นผลบนจอภาพเอง) และเพื่อให้การแสดงผลถูกต้อง เราจึงต้องไปเปลี่ยนค่าตัวแปรสองตัวคือ Xinc และ Yinc ของเดิมให้เหมาะกับคำสั่งการวาดรูปสี่เหลี่ยมด้วยคือเปลี่ยนเป็น Xinc:=32; Yinc:=24; โดยเราอาจเพิ่มคำสั่งตรวจค่า x, y ด้วยก็ได้เช่น

```
If x mod 8< > 0 then x :=
x + (8-x mod 8);
  If y mod 8 < > 0 then y : =
y + (8-y mod 8);
```

จะทำให้ความผิดพลาดในการแสดงผลไม่มีทางเกิดขึ้นได้เลย การเปลี่ยนแปลงนี้อยู่ในชุดคำสั่งของรูปที่ 3 คือชุดคำสั่ง OutTextBig การนำชุดคำสั่งในรูปที่ 11 ไปแทนรูปที่ 2 แล้วเก็บไว้ในไฟล์ ASCII3.PAS ก็จะได้ไฟล์ที่สร้างอักขระโดยเทคนิคการวาดรูปสี่เหลี่ยมตามต้องการส่วนรูปที่ 12 แสดงให้เห็นข้อความที่ได้จากวิธีวาดรูปสี่เหลี่ยมจะเห็นว่าช่องไฟจะกว้างกว่าเดิม ระยะห่างระหว่างสระกับอักษรกลางห่างมากขึ้น หากประสงค์จะให้มีผลดีกว่านี้ก็คงต้องแก้ไขไฟล์ข้อมูลอีกเล็กน้อยก็จะได้รูปแบบอักขระตามต้องการ การแสดงแบบนี้จะเป็นการแสดงแทนที่ภาพ ณ ตำแหน่งเดิมด้วย ส่วนวิธีไหนจะมีความไวในการแสดงผลมากกว่ากันผู้อ่านก็คงต้องสังเกตเองบนจอภาพ

พบกันใหม่คราวหน้าด้วยโปรแกรมป้ายประกาศ หรือ THAI BANNER ซึ่งขอบอกไว้ล่วงหน้าเลยว่าน่าสนใจมาก ๆ.....